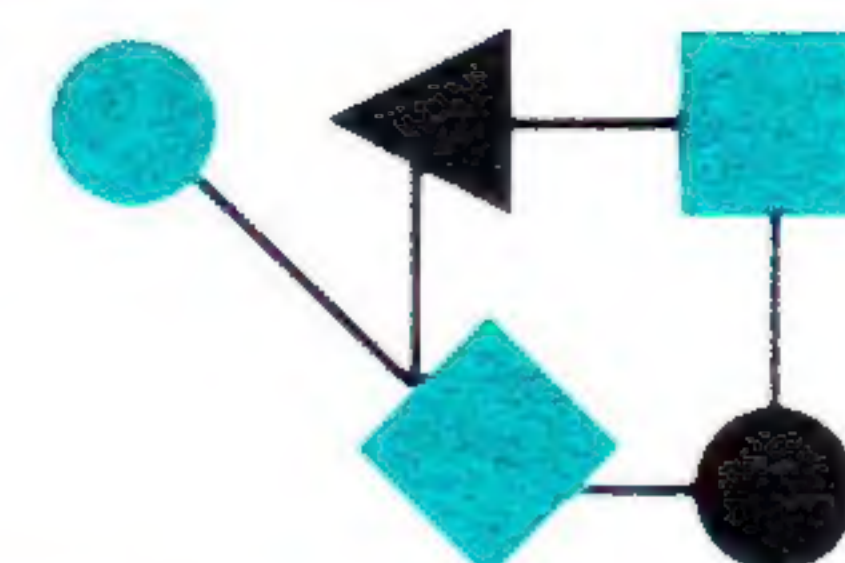


# CONNEXIONS



## The Interoperability Report

July 1988

Volume 2, No. 7

*ConneXions -  
The Interoperability Report  
tracks current and emerging  
standards and technologies  
within the computer and  
communications industry.*

### In this issue:

TCP Connection Initiation....	2
Net Management Update.....	12
The Enterprise Networking Event.....	13
Information on TCP/IP.....	14

ConneXions is published by Advanced Computing Environments, 480 San Antonio Road, Suite 100, Mountain View, CA 94040, USA. Phone: 415-941-3399.

© 1988  
Advanced Computing Environments.  
Quotation with attribution encouraged.

*ConneXions-The Interoperability Report*  
and the *ConneXions* masthead are  
trademarks of Advanced Computing  
Environments.

ISSN 0894-5926

### From the Editor

This month we feature an article on the details of a TCP connection initiation. As you will discover, a large number of steps take place between the time you see "Trying...." and "open" on your screen during a Telnet session (and several things may go wrong in the process). The article is by Greg Minshall of Kinetics, Inc.

As reported in our June issue, three groups have been tasked with developing Network Management standards for TCP/IP networks. On page 12 we have a brief status report from these groups.

The OSI protocol suite is becoming more real every day. The *Enterprise Networking Event* held in Baltimore in June provided an opportunity to witness the complete commitment by the vendors to providing OSI products in the near future. Several important interoperability demonstrations were shown. Our president, Dan Lynch was invited to speak (about TCP/IP!), and on page 13 he gives some impressions from the show.

In response to many requests I have compiled a list of books, documents, etc. giving information about TCP/IP. In future issues of *ConneXions* we will be reviewing some of the books.

The TCP/IP protocol suite uses 32-bit addresses. These IP-addresses are divided into a network part and a host-in-network part. The network part is called a *network number*. The Network Information Center (NIC) at SRI International can assign any organization using the TCP/IP protocol suite a unique IP network number. To request a network number assignment send a message to: HOSTMASTER@SRI-NIC.ARPA or call 800-235-3155 or 415-859-3695.

There are two steps to number assignment. First is finding out if you know what you are doing, and that you really need a number assigned. If the NIC determines that you do, a number will be assigned. The second part is determining if you are authorized i.e. "Are you sponsored or authorized by one of the government agencies that sponsor the Internet to connect to it?". For many applicants these two steps can be resolved at once. However, any organization using TCP/IP may have a "non-connected" network number assigned. If at a later time the organization becomes authorized to participate in the Internet they continue to use the same number, and supply the appropriate information to the NIC to have their number changed in status to "connected". It is very desirable to have a network number assigned by the NIC. If the NIC assigns you a number no other organization will be assigned that number.

## TCP Connection Initiation - An Inside Look

by Greg Minshall, Kinetics, Inc.

### Introduction and notation

The following is a discussion of the *specifics* of how a TCP connection between processes on two machines is initiated. This discussion is at a level appropriate for an intermediate level TCP discussion. The person with little or no familiarity with TCP will almost certainly be lost; the TCP literate will almost certainly find better things to do than read this discussion. However, for the person with a modest amount of TCP knowledge, and a desire to see “how it all fits together”, this article may prove useful.

A TCP connection progresses through certain states, and we will indicate states by using the notation “[stateA, stateB]”. The first state is the state as perceived on the A machine; the second state is the state as perceived on the B machine. State transitions will be denoted by “[oldA, oldB] --> [newA, newB]”. At the start of the process, it is assumed that the connection is closed on one machine and listening on the other; thus the state is:

[CLOSED, LISTENING]

While this is not intended to be a tutorial on TCP per se, we will remind ourselves that within the TCP/IP protocol suite an application's address is composed of a triple: the (or a) IP address of the machine on which the application is running; the protocol type of the transport layer (normally the value corresponding to either “TCP” or “UDP”)<sup>1</sup>; and the port number within the specific transport entity on the specific machine on which the application is running. For this reason, we will use the notation “IPa/TCP/PORTa” to denote “the application which is accessible via port ‘PORTa’ of the TCP transport protocol on the machine with IP address ‘IPa’”. We will use similar schemes to discuss other protocol entities within various systems<sup>2</sup>.

### Prelude

We begin with a user at machine with address *IPa*. The user is sitting at a terminal, and types the command:

**telnet violet.cchem.berkeley.edu.**

We notice that at this point the user is demonstrating some pre-knowledge. First of all, the user knows the name of the telnet command. This may seem like an obvious requirement; however, many of the current “Icon oriented—point and click” operating systems (e.g. Apple's Macintosh OS) obviate the need for knowing the name of a command. Second, and more germane to this discussion, we see that the user knows the host name of the remote system. In another type of network, one supporting, say, a generalized resource/service location protocol, the user might have presented to him/her the names of those hosts currently accepting connections. At any rate, let us assume the user now hits the RETURN key, causing the local system to execute the command.

<sup>1</sup> In reality this is not the “transport identifier”, but rather the “IP client identifier”. This is because not all clients of IP are easily classifiable as transport protocols.

<sup>2</sup> There is a large amount of abuse of notation going on here (and throughout this article); we console ourselves with the *thought* that all misused terms are in a one-to-one relation with the correct terms.

## Name to address translation

The telnet command begins execution. The first order of business is to translate the host name **violet.cchem.berkeley.edu** into a 32-bit *IP address*. There are two methods currently in use for performing this translation. The first is to look up the name in a local, static data base.

This method relies on a more-or-less constant network configuration and on the ability to acquire an approximate list of names and addresses. If this is the method in use on IPa, the translation proceeds independent of the state of the network and will only fail if the name **violet.cchem.berkeley.edu** is not found in the local database (eg, the user entered the wrong name, or the database is out of date).

## Domain Name System

The second method currently in use (and the preferred method) is known as the *Domain Name System* (DNS). (A summary of DNS is presented in *ConneXions* Volume 1, No. 6, Oct. 1987). If the telnet program at IPa uses the Domain Name System to translate **violet.cchem.berkeley.edu** into an IP address, the distributed data base is accessed. It is possible that IPa has, in the recent past, translated **violet.cchem.berkeley.edu**. If this is true, the translation may still be marked valid in IPa's local *name cache*. In this case, the translated address (which we will denote *IPb*) is returned.

Let us assume that the name-to-address translation is *not* contained in the local cache. In this case, if IPa is *near* (in an administrative sense) to the target name, then possibly only one name server need be contacted. If IPa is *far* from the target name, then the request may go various places around the network. For example, possibly one of the root nameservers will need to be queried. The root nameserver will be asked to translate **violet.cchem.berkeley.edu**. It will, undoubtedly, deny knowledge of the offered name, but will volunteer the address of one or more nameservers for **.edu**.

Similarly, a nameserver for **.edu** will be asked to translate the target name, but will indicate ignorance plus the address of one or more nameservers for **berkeley.edu**. Finally, proceeding down the tree, the telnet command (or, more correctly, a *name resolver* invoked by the telnet command) will eventually discover an address for the target name; or discover (from an *authoritative name server* for that portion of the name space in which the target name is located) that the target name is invalid (ie: has no mapping); or will be unable to conclude the translation process because of an inability to contact the next nameserver. In the last two cases the telnet command will report an error to the user.<sup>3</sup>

If the name to address translation succeeds, the telnet command has determined an IP address which should correspond with the name **violet.cchem.berkeley.edu**. Let us call this address *IPb*. Note, by the way, that there is no defined way of ensuring absolutely that, in fact, *IPb* corresponds to **violet.cchem.berkeley.edu**.

<sup>3</sup> Note that the two cases should be reported in a different manner to the user; unfortunately, many applications report "host unknown" in the face of *any* error in the translation process.

## TCP Connection Initiation (*continued*)

### Well-known socket "Telnet"

Next, telnet needs to determine the well-known socket number for the telnet service in order to form the application address of the telnet service to be contacted. This may be hard-coded into the program (or compiled in from information in an include file), or this may require access to some local database (eg: `/etc/services` on Berkeley UNIX systems). Let us call this well-known socket number *PORTtelnet*.<sup>4</sup>

### Establishing a local TCP Control Block

At this point, the telnet command requests that the local TCP (IPa/TCP) allocate a socket for use in establishing a new connection. This request may fail if, for example, there is a resource shortage on the local system (not enough network buffers, for example) which would prevent the allocation. Assuming, however, that the request succeeds, the telnet command then requests a connection to IPb/TCP/PORTtelnet. In setting up this connection (or even in allocating space for the local socket), IPa/TCP assigns a port number to the socket which the telnet command is using. Let us call this port *PORTa*. So, telnet is trying to establish a connection between IPa/TCP/PORTa and IPb/TCP/PORTtelnet.

### Three-way Handshake

As an overview, let us describe what protocol information IPa/TCP and IPb/TCP need to exchange to set up the connection. They need to send each other a SYN bit; they need to acknowledge the other's SYN bit; and the connection is set up. The SYN bits indicate a desire to initiate a connection; the SYN bits, with their subsequent acknowledgements, *synchronize* the two TCP's as to the state of the connection. This series of protocol exchanges is known as a *Three-way Handshake* (because IPa/TCP sends a SYN; IPb/TCP sends a SYN and an ACK together; and IPa/TCP sends an ACK). This is conceptually simple, but a bit involved when we follow the various control and data paths.

### The first datagram

First of all, IPa/TCP calculates an *Initial Send Sequence* number for the connection. We will call the number calculated *ISSa*. The correct determination of this number is essential in protecting this new connection from being fooled into accepting data (or control messages) which may have been delayed in the network from some previous connection.

Next, a datagram with the sequence number "ISSa", and containing the SYN bit, is prepared. The SYN bit, which indicates a desire to establish a new connection, is protected by the TCP sequence number mechanism, so ISSa is the sequence number of the SYN bit. An acknowledgement number (from IPb/TCP) of ISSa+1 will acknowledge IPb/TCP's receipt of the SYN bit.

IPa/TCP now checksums the datagram and presents it to IPa for delivery to IPb. At this point, a state transition occurs:

[CLOSED, LISTENING] --> [SYN SENT, LISTENING]

<sup>4</sup> Technically, a *port* is a transport layer (TCP or UDP) access point, with no indication of what machine the port is on; a *socket*, on the other hand, defines a specific access point, i.e: the instantiation of a port on a particular machine. Thus we should *really* be speaking about "well known ports"; however, "well known sockets" is the commonly used term.

IPa examines the value "IPb". It is possible that IPa equals IPb, or that IPb is a second address for the machine on which IPa is running. In either of these two cases the datagram is presented to the local TCP (IPa/TCP, which is IPb/TCP in this case).

Let us assume, however, that IPb is determined to be on a separate machine. IPa now prepends an IP header to the datagram, using information passed by IPa/TCP: the destination IP address (IPb), the source IP address (IPa), and the protocol type (TCP). Additionally, IPa fills in other fields in the IP header. IPa then performs a checksum on the IP header portion of the datagram. This checksum protects *only* the IP header, ie: the IP control information contained within the datagram. This checksum is checked (and, actually, recomputed) by each IP router through which the datagram passes, as well as by the end system (IPb).

IPa next tries to determine if IPa and IPb share a common physical link (e.g., they are both on the same Token Ring network or the same Ethernet cable). If this is true, then the datagram should be presented to *DLa* (the data link layer in the machine in which IPa is resident) for delivery to *DLb*.

#### Routing via a gateway

If IPa and IPb do *not* have a common physical link, then IPa will need to send the datagram to an *Intermediate System* (IS) for forwarding to IPb. (IS is ISO terminology; *IP Router* and *IP Gateway* are two names given to these intermediate systems within the TCP/IP community). Through some management or administrative process, IPa will have a list of such ISs to use when making routing decisions. IPa will go through a *route lookup*, which should result in an IP address which will be the next hop for the datagram in question (by definition, this next hop has a physical link in common with DLa). If IPa is unable to find an IS which should be the next hop, it will report an error to IPa/TCP. Otherwise, assume that the determined next hop is IPc. In this case, IPa will deliver the datagram to DLa, requesting that the datagram be sent to IPc.

Note that in this case the datagram (which is flowing via IPc) looks exactly like the datagram which would flow to IPb if IPb were in fact accessible via a common physical link. The only difference is that the data link layer will encapsulate the data link packet with a data link address corresponding to IPc if there is an intermediate hop, or with a data link address corresponding to IPb if there is a direct connection.

#### Address resolution by the Data Link Layer

To simplify things, let us call the next hop, be it IPb or IPc, *IPd*.

IPa delivers a datagram to DLa, requesting that it be delivered to IPd. DLa now has a problem. DLa knows that the datagram is to be delivered to IPd. However, DLa talks with other machines sharing the same physical link by means of data link addresses—not IP addresses. For example, on Apple's LocalTalk Network data link addresses are one byte; while on an IEEE 802.5 (Token Ring) they are (in the case interesting to us) 6 bytes in length.

DLa needs to use some mechanism to translate IPd into DLd, the data link address of the machine corresponding to IPd. There are two schemes in common use to perform this translation.

*continued on next page*

## TCP Connection Initiation (*continued*)

In some cases (such as the Apple's LocalTalk), we may easily map the *host portion* of the IP address IPd *onto* the data link address space.<sup>5</sup>

Since we have an onto map, we may use the host portion of the IP address (or a portion thereof) "as" the data link address. Thus, we *extract* the data link address directly from the IP address. Note, however, that when IP subnetting is in use, the host portion of an IP address may be "arbitrarily small." Thus, this method may not be suitable in a subnetted environment.

**ARP** In the second scheme (used by IEEE 802.2 networks, including both Token Ring and Ethernet networks), the data link address space is larger than the IP address space (or, in particular, than the host portion of the IP address space), and there is no way of defining an a priori mapping of the IP address space *onto* the data link address space (and, of course, one needs an *onto* map since there is no way of guaranteeing that data link addresses will fall within a certain range). In this latter case, a protocol known as *Address Resolution Protocol* (ARP) is often invoked to perform the translation between IP addresses and data link addresses.

ARP is a protocol which relies on the *broadcast nature* of the data link layer on which it runs. On most local area networks, for example, each data link entity recognizes one special data link address as meaning "receive and process at all stations;" this data link address is known as the *data link broadcast address* (or "Ethernet broadcast address," or "IEEE 802.2 broadcast address"). ARP creates request packets requesting a translation from a network layer (IP, in this case) address into a data link layer address and broadcasts them over the physical link. If an ARP entity which *knows* the correct translation receives the request, it replies with the desired translation.

DLa will request DLa/ARP for a data link address corresponding to IPd. DLa/ARP will first look in a cache of recently translated IP addresses to see if a valid entry for IPd is to be found. If such a valid entry is found, DLa/ARP returns the value "DLd" which was previously discovered (or administratively assigned) to be the data link address for IPd.

If no valid entry is found in the ARP cache, DLa/ARP will create an *ARP request packet* which indicates a desire to translate an IP address (IPd) into a data link address. Each network layer protocol which is translated by ARP has a specific type value associated with it. The packet will indicate that this is an attempt to map from an IP address.

Additionally, each type of data link layer which uses ARP (ie: IEEE 802.5, Ethernet, etc.) is assigned a unique data link type value. The packet will indicate an attempt to map into the data link type of DLa.

<sup>5</sup> An *onto map* is a mapping from members of one set (the *domain*; in this case, the host portions of IP addresses) to members of a second set (the *range*; in this case data link addresses) such that for each possible member in the range there is at least one member in the domain which maps into that member of the range

DLa/ARP will now deliver this packet to DLa to be broadcast over the connected physical layer. DLa will broadcast the packet as directed. The original packet from IPa to IPb via IPd may be discarded by DLa/ARP (it will be re-generated by IPa/TCP after a retransmission timeout), or, in a more advanced implementation of ARP, the packet may be held to be sent as soon as an answer to the ARP request is received. If, after some period of time, no response is received to the ARP request packet, DLa/ARP will discard the saved packet.

Note that this processing by ARP is happening with the complete ignorance of IPa/TCP and of IPa. This is beneficial in keeping down the size of the (logical) interface between IPa/TCP and IPa, and that between IPa and DLa. However, this ignorance means that there is no way IPa/TCP can determine that it is the translation process which is (or may be) timing out, rather than the network delay being very long (or packets being dropped in an unreliable network). If this sort of knowledge *could* flow back up to IPa, for example, and if IPd was an intermediate system, we could decide that IPd is no longer a valid route for *any* outgoing connection (since we are unable to contact IPd at the data link level). IPa would then have the chance of allocating an alternate route to IPb. However, in most current implementations, the route to IPd might never be deleted if IPd was able to *send but not receive* packets. These various “un-architected but useful” control flows are under discussion by various groups of people interested in TCP/IP (and, in general, network) protocol design.

#### Back to the datagram

Now, an ARP request packet is flowing from DLa to the data link broadcast address. Each data link entity will examine the broadcast packet. Each entity with ARP capability will deliver the incoming packet to the ARP processing module in their node.

DLd (i.e.: the data link entity co-resident with IPd) will examine the packet and forward it to its ARP processing entity. DLd/ARP will determine, via some local function, that DLd is a valid data link address for IPd. DLd/ARP will then prepare an ARP response packet. This response packet will contain the information that came in the request packet (this is to allow DLa/ARP to correlate the response with its original request), plus the value “DLd,” the data link address of DLd. At this same time, DLd/ARP *may* decide to enter the mapping (IPa, DLa) into its own local ARP cache. This is a quite reasonable policy, given that *most* IP traffic flows are bi-directional (and IP is the main client protocol for which ARP operates at present).

DLd/ARP will then deliver the ARP response packet to DLd for transmittal to DLa. DLd will attempt to do so. Any errors in this attempt will probably be ignored by the ARP mechanism.

DLa will receive the ARP response packet from DLd. Noticing that the packet is of the ARP type, the packet will be handed to DLa/ARP for processing. DLa/ARP will discover that it now knows the mapping from IPd to a data link address. If it has saved the original packet from IPa to IPb via IPd (or a re-transmitted one, if the ARP process has taken a “long” time), this packet will be delivered to DLa for transmission to DLd. In the case where the original packet has not been saved, it will soon be re-transmitted by IPa/TCP.

*continued on next page*

### TCP Connection Initiation (*continued*)

DLa now transmits the packet (containing the TCP/IP datagram from IPa to IPb via IPd) to DLd.

We now turn our attention to DLb, the data link layer co-residing with IPb. DLb will eventually receive the original packet (either from DLa if IPa and IPb share a common physical link, or from some intermediate system providing IP routing services). DLb will examine the network protocol type field in the incoming packet (this corresponds loosely to the *Destination Service Access Point* for IEEE 802.2 networks), and will hand the packet to IPb (if IPb does not exist, the packet will be silently discarded).

IPb first verifies the IP header checksum contained in the datagram. If the checksum is incorrect, the datagram will be silently discarded.

If the IP checksum is valid, IPb will examine the transport protocol type field of the incoming IP datagram. If there is no IPb/TCP (ie: IPb does not have an associated TCP processing entity), an *Internet Control Message Protocol* (ICMP) error message will be sent to IPa/ICMP, informing IPa/ICMP (and IPa/TCP, if IPa/ICMP shares the information with IPa/TCP) that there is no TCP located at IPb.

If IPb/TCP does exist, IPb will pass the incoming datagram to IPb/TCP.

#### TCP Checksum

IPb/TCP will examine the incoming datagram. The first act will be to validate the *TCP checksum* of the datagram. This checksum protects the TCP control information in the datagram as well as application data in the datagram (additionally, the TCP checksum protects, somewhat redundantly, certain information from the IP header). If the checksum in the incoming datagram is invalid, the datagram is discarded.

If the TCP checksum is valid, IPb/TCP will examine the contents of the TCP header of the datagram. Recall that this datagram is from IPa/TCP/PORTa to IPb/TCP/PORTtelnet, contains sequence number ISSa, and has the SYN bit turned on. IPb/TCP will recognize this datagram as an attempt to initiate a *connection* (because of the presence of the SYN bit), and will check to make sure that the connection does not already exist. A TCP connection is identified by the tuple (IPa, PORTa, IPb, PORTtelnet) (in this case), and so IPb/TCP will search its list of active connections looking for (IPa, PORTa, IPb, PORTtelnet).

#### Half-open connections

If such a connection (known as a *half-open connection*) is found, IPb/TCP will send an acknowledgement datagram for the *already existing connection* to IPa/TCP, and will drop the incoming datagram.<sup>6</sup>

<sup>6</sup> This could happen if, for example, there *had* existed a connection (IPa, PORTa, IPb, PORTtelnet) in the past, and IPa/TCP had restarted without IPb/TCP's notice. In this case, the current "state" of the connection would be [SYN SENT, ESTABLISHED], rather than [SYN SENT, LISTENING].

This acknowledgement datagram will acknowledge all already received data on the existing connection. IPb/TCP does this because the received datagram (with the SYN bit) from IPa/TCP might be a very old datagram which was delayed in transit by the network. IPb/TCP does not wish to interfere with a possibly functional connection because of a spurious datagram, so it merely informs IPa/TCP of the existence of the existing connection.

### Closing half-open connections

In the case where the SYN datagram *was* indeed an old (or, possibly, corrupted) datagram, IPa/TCP will, to all intents and purposes, ignore the acknowledgement datagram. However, in the case we are describing, IPa/TCP will know that *it* has no existing connection (IPa, PORTa, IPb, PORTtelnet) at the sequence number which the datagram from IPb/TCP is acknowledging (this is why the correct determination of the Initial Send Sequence number above is so important for TCP robustness). So, IPa/TCP will send a *TCP reset* datagram to IPb/TCP. This reset datagram will contain as a sequence number the acknowledgement number of the datagram from IPb/TCP (i.e. of the existing connection).

This reset datagram will cause IPb/TCP to tear down the old connection, synchronizing both TCPs in preparation for creating the new connection.

### Normal processing

Let us assume, however, that IPb/TCP scanned its list of active connections without finding (IPa, PORTa, IPb, PORTtelnet). In this case, IPb/TCP will look to see if there is any process listening on PORTtelnet. If there is no such process, IPb/TCP will send an ICMP message to IPa/ICMP informing IPa/ICMP (and, possibly, IPa/TCP) that no such port is available for connection at IPb/TCP. Then, the incoming datagram will be discarded.<sup>7</sup>

If, however, there is a process listening for incoming connections on IPb/TCP/PORTtelnet, IPb/TCP will construct a datagram to send in reply to the received datagram. This datagram will contain an acknowledgement number of ISSa+1 (acknowledging "all" data sent on the connection before ISSa+1), will contain a SYN bit of its own, and will be sent with a sequence number of ISSb (ie: the Initial Send Sequence number calculated, independent of the value of ISSa, at this point in time by IPb/TCP). This datagram from IPb/TCP to IPa/TCP assures IPa/TCP that IPb/TCP has heard IPa/TCP's request to initiate a connection, and requests IPa/TCP to confirm receipt of IPb/TCP's willingness to establish a connection.

IPb/TCP will now deliver the datagram to IPb for delivery to IPa. At this point, a state transition occurs:

[SYN SENT, LISTENING] --> [SYN SENT, SYN RECEIVED]

In a manner similar to IPa previously, IPb will examine the address IPa, attempting to determine how to send the datagram. If IPa equals IPb, or if IPa and IPb are two addresses for the same machine, the datagram will be delivered to the TCP processing entity on the local machine.

<sup>7</sup> In this case our *initial* state would have been [CLOSED, CLOSED], rather than [CLOSED, LISTENING].

### TCP Connection Initiation (*continued*)

Otherwise, IPb will look to see if (based on properties of IP addresses) IPa and IPb share a common physical link. If this is the case, IPb will deliver the datagram to DLb for delivery to IPa.

If not, IPb will attempt to determine which intermediate system should be the *next hop* for datagrams addressed to IPa. It is not *too* unusual for IPa to know a route to IPb while IPb does *not* have a route to IPa. If this is the case, IPb will discard the datagram (and the state will remain [SYN SENT, SYN RECEIVED] until one side or the other aborts the connection; eventually, because of the fact that neither SYN is being acknowledged, both should do so). Again, this is quite often done without the knowledge of IPb/TCP (and certainly without the explicit knowledge of IPa/TCP!). However, if a route to IPa *is* found, the datagram is delivered to DLd with instructions to deliver it to the appropriate intermediate system. It is important to note, at this point, that the route *from* IPa to IPb may not involve the same set of intermediate systems that the route from IPb *to* IPa does; this is a property inherent in *connectionless* network layers (and even in some network layers which operate in a connected mode).

Let us assume that the datagram from IPb/TCP to IPa/TCP is delivered to DLb with instructions to deliver it to IPe. (IPe may equal IPa if, for example, IPb and IPa share a common physical link).

DLb will request DLb/ARP, in an ARP-based environment, to provide a translation from IPe into a data link address. ARP will proceed as before. This time, however, the chances that the mapping from IPe into a data link address already exists in the local ARP cache is much larger than when we started at DLa looking for a translation for IPd. This is because it is quite likely that IPe was, in fact, the IP entity which delivered the original datagram from IPa/TCP to IPb/TCP. In this case, IPe likely performed an ARP lookup to determine DLb from IPb, and more than likely DLb/ARP saved the mapping information (IPe, DLe).

#### Connection Established

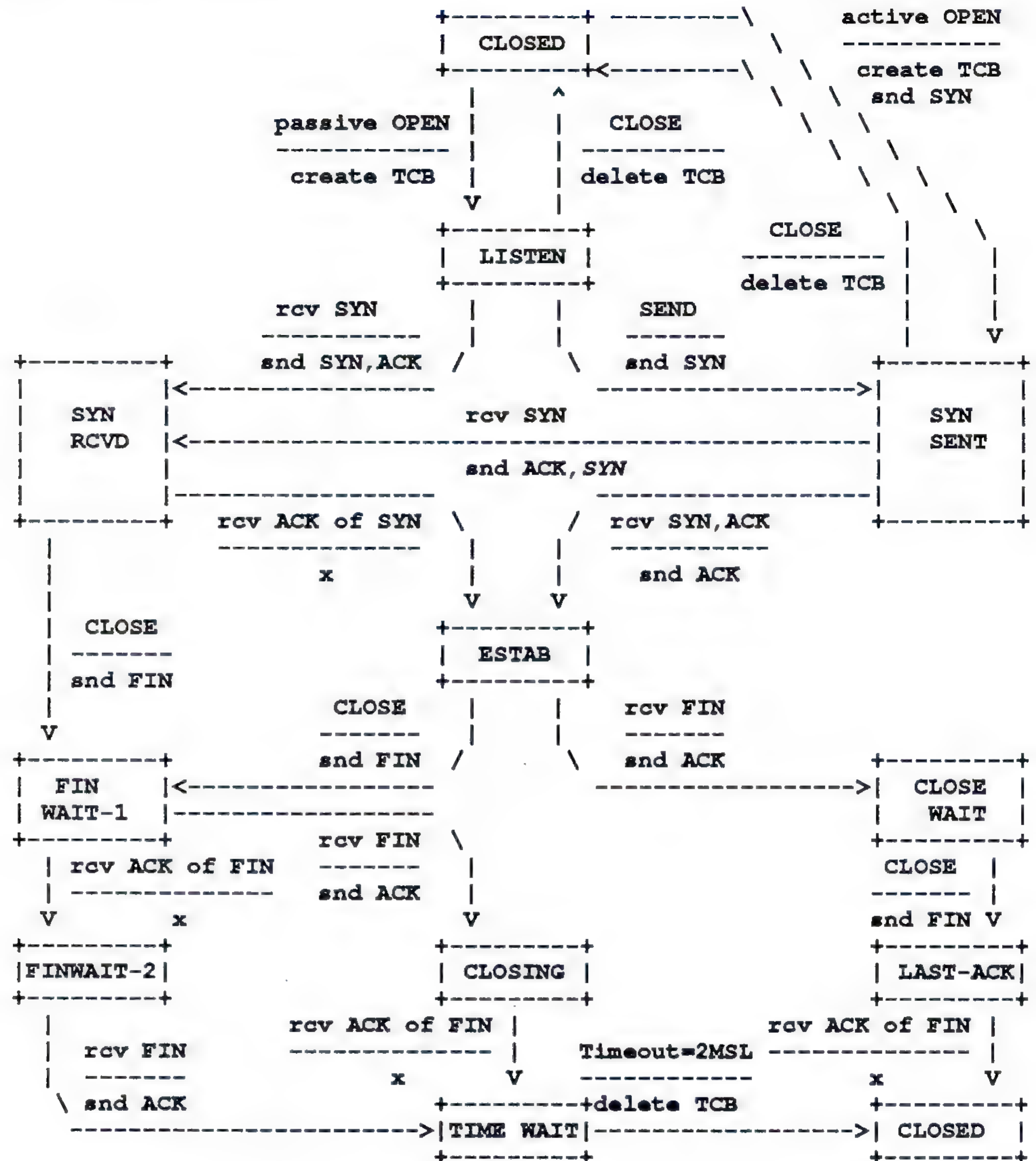
Eventually, the datagram from IPb/TCP to IPa/TCP will arrive at IPa/TCP. IPa will deliver the incoming datagram to IPa/TCP. IPa/TCP will verify the TCP checksum of the incoming datagram, and will then determine that the acknowledgement number of this datagram is correct (ie: is ISSa+1). Assuming this is all correct, IPa/TCP will send a datagram to IPb/TCP with an acknowledgement number of ISSb+1 (i.e. acknowledging IPb's SYN bit). IPa/TCP will then place the connection in the ESTABLISHED state and will allow application data to flow on the connection. At this point, a state transition occurs:

[SYN SENT, SYN RECEIVED] --> [ESTABLISHED, SYN RECEIVED]

IPb/TCP will receive the acknowledgement packet from IPa/TCP, and will likewise place the connection in the ESTABLISHED state and allow application data to flow. Another state transition occurs:

[ESTABLISHED, SYN RECEIVED] --> [ESTABLISHED, ESTABLISHED]

After these (possibly) lengthy acrobatics, the connection is open and the user is now able (via the telnet protocol) to log in to **violet.cchem.berkeley.edu**.



TCP Connection state diagram (from RFC 793) \*

## Acknowledgements

I would like to thank Messrs. Pat Flaherty, Bill Northlich, and Mark Wittenberg, all of Kinetics, for their technical and grammatical advice, and Mr. Cliff Frost, of UC Berkeley, for his technical advice and discussion of the overall form of this article. Notwithstanding these acknowledgements, all defects in this article (of both form and content) are attributable to me.

**GREG MINSHALL** is a programmer with Kinetics, a division of Excelan Inc., specializing in providing networking products for Apple Macintosh computers. Greg also spends a small portion of his work time at UC Berkeley.

\*Ed.: There is an error in the state diagram in RFC 793, the picture above shows the *corrected* version. The box labelled "SYN SENT" should have an arrow going to the left with "snd ACK, SYN," below it. In RFC 793 this is shown as "snd ACK" only.

## Network Management Update

As mentioned in our June issue, network management in the Internet community is being developed by three working groups according to the IAB's recommendations, as outlined in RFC 1052.

### **MIB Working Group**

*The MIB Working Group*, chaired by Craig Partridge, BBN Laboratories, was charged with developing a proposal for a standard TCP/IP Management Information Base (MIB). A MIB defines how a system must be instrumented, and how those instrumented values are made visible to the management protocols. The MIB group met in April and May and developed two documents. A Structure of Management Information (SMI) which defines the data types used in the MIB, and a proposed MIB itself (of a little over 100 parameters). These documents were then circulated for comment until mid-June. The final version of these documents will be released as RFCs soon.

### **SNMP Extensions Working Group**

*The SNMP Extensions Working Group*, chaired by Marshall T. Rose, The Wollongong Group, Inc., was charged with aligning the Simple Network Management Protocol (as specified in IDEA11) with the output of the MIB Working group. This work has recently been completed, and a draft document has been made available for comment. Both the University of Tennessee at Knoxville and NYSERNet, Inc., have interoperable prototypes of the new system running, and expect to have production implementations completed before the end of the summer.

### **Netman Working Group**

*The Netman Working Group*, chaired by Lee LaBarre, Mitre Corp., consists of about a dozen vendors who are developing common protocols and database definitions for network management using the OSI protocols (CMIS/CMIP) as their basis.

### **September Demo**

A subset of the Netman Group, chaired by George Marshall, 3Com/Bridge, has been working to develop a prototype network management system (the "September Demo") to be shown at INTEROP 88, the next TCP/IP Interoperability Conference & Exhibition. The group has been meeting regularly this spring and summer and has developed a set of implementor's agreements for the demo. In July, an Implementor's Workshop will be held at the Unisys facility in Santa Monica, California. At this time, the participating vendors will be briefed on the Central Demo Manager software which has been developed by Unisys and Mitre. Object code will be distributed to the participants in order to perform interoperability testing with the Agents developed by the other vendors. The major part of September will be taken up by demonstration staging activities at the 3Com/Bridge division in Mountain View, California. During this time, extensive interoperability testing between the participants' implementations will be performed.

## The Enterprise Networking Event

by Dan Lynch

The much awaited OSI networking event finally took place the second week of June. I regarded it as a big success. It consisted of a 3 day conference, with many tutorials, and a very complex exhibit.

### Conference

The conference was primarily aimed at technical managers who need to learn how OSI works, how it is progressing in the standards arena, and what products are available. There were also a number of talks centering on TCP/IP to OSI transition methods. Questions were asked that indicate that people are anxious to get started with multivendor networking and in the cases where there are not enough OSI products available the questioners wanted to know about the risks involved in going with TCP/IP today. At this juncture it is apparent that there are more solutions to this problem than one would imagine. My own talk suggested that the ISODE approach of letting users run the OSI applications on their existing TCP/IP networks was the quickest way to get the benefit of OSI without disrupting existing practises.

### Exhibits

The Exhibits part of this event were very impressive. They were not just the usual trade show approach. Instead there were 9 huge booths each containing activities that incorporated products from 5-10 vendors. As an example, the Air Force/Industry booth had a group of businesses that took a large airplane machining project and broke it up into a number of subcontracts. Each subcontractor ran its own applications (task assignment, initial design, contract administration, negotiation, customer byoff, machining, shipping, etc.) on the computers of their choice. They passed the work documents around using X.400 and FTAM within the booth and around the floor and out of the building to foreign subcontractors. It all worked. And it was *fast*.

### Practical uses

The networking part of all this looked very smooth. What impressed me the most was seeing that industry is learning how to do business by using networks to pass the technical and administrative data around in seconds instead of days and weeks. The payoff in efficiency is huge. Sometimes we technologists forget that folks plan to use the fruits of our labor to do things that have nothing to do with networking per se. That is a good sign.



## Information about TCP/IP

We are often asked about sources for information on TCP/IP protocol documentation, books, product listings etc. In this article we will give an overview of what is available.

### Books

For general information about computer networking see: *Computer Networks*, by Andrew S. Tanenbaum published by Prentice-Hall, ISBN 0-13-164699-0. Specific books on TCP/IP include:

- *Internetworking With TCP/IP—Principles, Protocols, and Architecture*  
by Douglas E. Comer, Prentice-Hall, ISBN 0-13-470154-2
- *An Introduction to TCP/IP*  
by John Davidson, Springer-Verlag, ISBN 0-387-96651-X
- *Handbook of Computer-Communication Standards  
Volume 3: Department of Defense Protocol Standards*  
by William Stallings et al, Macmillan, ISBN 0-02-948072-8

### RFCs and Military Standards

Request For Comments (RFCs) contain all the protocol specifications used by the Internet community. Note that only some of these are “Official Internet Protocols”. (The list of official protocols is itself an RFC and is published periodically). See also RFCs 999, 1000, and 1012 for a guide to RFCs. There are five (5) Military Standard (MIL-STD) documents. The IP and TCP MIL-STDs are completely different from the corresponding RFCs yet intending to define the same protocols. The FTP, MAIL, and Telnet MIL-STDs are copies of the corresponding RFCs. Implementors should always use all the available information. There are some known errors in MIL-STDs (specifically, see RFC 963 and RFC 964). In cases of confusion, the RFCs should take precedence over MIL-STDs.

### Getting RFCs

RFCs are available from the DDN Network Information Center at SRI International. Hardcopies can be ordered by calling 800-235-3155 or 415-859-3695. Online versions may be copied via anonymous FTP from the RFC: directory on host SRI-NIC.ARPA (10.0.0.51 or 26.0.0.73). (Check out the file RFC:RFC-INDEX first). If you are outside the “connected Internet” and don’t have access to FTP, you can still receive RFCs by simply sending electronic mail to SERVICE@SRI-NIC.ARPA. Put the RFC number in the “Subject:” field (e.g., Subject: RFC 980). The DDN NIC also publishes the *DDN Protocol Handbook*, a three-volume set, which contains most of the important RFCs related to TCP/IP. Although this document is now a little out of date it is definitely worth getting for reference.

### Getting MIL-STDs

MIL-STDs are available from:

Naval Publications and Forms Center, Code 3015  
5801 Tabor Avenue  
Philadelphia, PA 19120  
215-697-3321 (order tape) 215-697-4834 (conversation)

### List of implementations

*The DDN Protocol Implementations and Vendors Guide*, also available from the DDN NIC, contains listings of vendor products for TCP/IP, X.25, and (a few) OSI products. This document is extremely useful for finding protocol software for your particular system.

The Vendors Guide is also handy if you're bold and have in mind doing a full-blown implementation of TCP/IP from scratch. You might find that someone already beat you to it, or perhaps a port of somebody else's software is a more appropriate avenue to pursue.

The NIC also has a handy bibliography on TCP/IP and ISO/OSI articles.

## Network Information Centers

Most of the the large networks which comprise the connected (and in some cases non-connected) Internet operate Network Information Centers which provide help and information for current or potential users. The major NICs are:

- DDN Network Information Center  
SRI International  
333 Ravenswood Avenue  
Menlo Park, CA 94025  
800-235-3155 or 415-859-3695  
NIC@SRI-NIC.ARPA
- NSF Network Service Center  
BBN Laboratories  
10 Moulton Street  
Cambridge, MA 02238  
617-873-3400  
NNSC@NNSC.NSF.NET
- CSNET Coordination and Information Center (CIC)  
BBN Laboratories  
10 Moulton Street  
Cambridge, MA 02238  
617-873-2777  
CIC@SH.CS.NET
- BITNET Network Information Center  
777 Alexander Road  
Princeton NJ 08540  
609-520-3377  
BITNET: INFO@BITNIC

## TCP-IP mailing list

A good source of information on TCP/IP is the electronic mailing list "TCP-IP@SRI-NIC.ARPA". The volume of messages on this list is quite high, and it would probably take you a year or more to read all the archives, but being on this list is very helpful if you are at all involved with TCP/IP networking. The mailing list is also distributed on USENET and is available as **comp.protocols.tcp-ip**.

## Tutorials and seminars

Courses on TCP/IP are available from Advanced Computing Environments. Tutorials are offered several times a year:

- In conjunction with our *TCP/IP Interoperability Conferences*
- As stand-alone Tutorial Series
- As in-house tutorials tailored to your specific needs

Additionally, we schedule special seminars (such as the recent *TCP Performance Seminar*).

Finally, the monthly newsletter you are reading now is designed to give timely information about the ongoing developments in the world of TCP/IP and ISO/OSI protocols.

CONNEXIONS

480 San Antonio Road  
Suite 100  
Mountain View, CA 94040

FIRST CLASS MAIL  
U.S. POSTAGE  
PAID  
SAN JOSE, CA  
PERMIT NO. 1

CONNEXIONS

PUBLISHER Daniel C. Lynch

EDITOR Ole J. Jacobsen

EDITORIAL ADVISORY BOARD Dr. Vinton G. Cerf, Vice President, National Research Initiatives.

Dr. David D. Clark, The Internet Architect, Massachusetts Institute of Technology.

Dr. David L. Mills, NSFnet Technical Advisor; Professor, University of Delaware.

Dr. Jonathan B. Postel, Assistant Internet Architect, Internet Activities Board; Division Director, University of Southern California Information Sciences Institute.

CONNEXIONS

Subscribe to CONNEXIONS

U.S./Canada \$100. for 12 issues/year \$180. for 24 issues/two years \$240. for 36 issues/three years  
International \$ 50. additional per year (Please apply to all of the above.)

Name \_\_\_\_\_ Title \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Country \_\_\_\_\_ Telephone ( ) \_\_\_\_\_

☐ Check enclosed (in U.S. dollars made payable to CONNEXIONS ).

☐ Charge my ☐ Visa ☐ Master Card Card # \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

Please return this application with payment to:

CONNEXIONS

480 San Antonio Road Suite 100  
Mountain View, CA 94040  
415-941-3399

Back issues available upon request \$10./each  
Volume discounts available upon request